Report: Assignment 3

Rajeev Gupta 2019CS51018

Part 1 Report: Controller Hub and Learning Switch

Objective: In this part, we compared the performance of a Hub Controller and a Learning Switch in our network topology.

Test Setup:

- Network Topology: Hosts `h1`, `h2`, `h3`, `h4`, and `h5` connected to switches `s1` and `s2`.
- Ryu Applications: Controller Hub (`controller_hub.py`) and Learning Switch (`learning_switch.py`).

Test Results and Analysis:

1. Conducting 3 Pings for Each Case:
    - We initiated 3 ping tests from various hosts to measure latency. The latency values were recorded and compared.

    Explanation:

    - In both scenarios, the Hub Controller and Learning Switch were tested for latency.
    - Latency Differences:
        - The Hub Controller showed higher latency because it redirects all traffic to itself and then forwards it to all switch ports except the incoming port. This process introduces additional delay.
        - The Learning Switch installed flow rules on the switches based on the MAC to Port mappings it learned from incoming traffic. This led to lower latency as traffic is forwarded more efficiently.
    - Differences Between `h2` and `h5`:
        - `h2` and `h5` experienced similar latency differences as described above for both controller types.

2. Throughput Test Between `h1` and `h5`:
    - A throughput test was conducted between `h1` and `h5` to measure data transfer rates.

    Explanation:

    - The Hub Controller showed lower throughput because of its method of handling traffic, which can lead to congestion and inefficiency.
    - The Learning Switch, with optimized flow rules, demonstrated higher throughput as it forwarded traffic based on learned MAC to Port mappings.
3. Running `pingall` in Both Cases:
    - We ran `pingall` tests to verify the installed rules on switches.

    Explanation:

    - The Hub Controller installed rules that redirected all traffic through the controller, resulting in a larger number of rules and potentially higher resource usage.
    - The Learning Switch installed rules based on the learned MAC to Port mappings, resulting in a more efficient rule set.

Part 2: Firewall and Monitor Report

Introduction

In Part 2 of the assignment, we continued to work with the network topology established in Part 1. The primary objectives of this part were to implement a firewall and monitoring system to enhance network security and track network traffic. This report provides an overview of the implemented firewall and monitoring functionality, along with answers to specific questions posed in the assignment.

Firewall Implementation

Firewall Rules

In this section, we discussed the firewall rules implemented in our Ryu application to restrict communication between specific hosts. The implemented rules are designed to prevent communication between pairs of hosts. We have made the assumption that communication between certain hosts poses a security risk, and our firewall rules aim to mitigate these risks.

Example Firewall Rules:

1. Block communication between `h1` (MAC: 00:00:00:00:00:01) and `h2` (MAC: 00:00:00:00:00:02).
2. Block communication between `h2` (MAC: 00:00:00:00:00:02) and `h3` (MAC: 00:00:00:00:00:03).
3. Block communication between `h4` (MAC: 00:00:00:00:00:04) and `h5` (MAC: 00:00:00:00:00:05).

Real-Time Firewall Policies

In the context of real-time firewall policies, our implementation allows for dynamic rule adjustments. We can modify and update firewall rules as network conditions change. This dynamic behavior ensures that pre-existing rules do not interfere with new policies. For example, we can add, modify, or remove rules in real-time to respond to evolving security requirements.

Monitoring Implementation

Packet Counting

We have implemented a packet counting mechanism in our Ryu application to monitor network traffic. This functionality tracks the number of packets originating from specific hosts on individual switches. The assumption here is that monitoring packets from specific hosts is essential for security and performance analysis.

Example Packet Counting:

- On `Switch 1 (S1)`, we count packets from `h3` (MAC: 00:00:00:00:00:03).
- On `Switch 2 (S2)`, we count packets from `h4` (MAC: 00:00:00:00:00:04).

Test Results and Observations

Firewall Testing

We conducted extensive testing to validate the firewall rules' functionality. During the testing phase, we observed that the implemented rules effectively restrict communication between hosts as intended. Pings and communication attempts between restricted host pairs were blocked, confirming the firewall's effectiveness.

Monitoring Validation

We validated the monitoring functionality by generating network traffic and observing the packet counting mechanism. The packet counts accurately reflected the number of packets originating from the specified hosts on their respective switches. The monitoring system is functioning as expected.

Part 3: Load Balancer Report

Introduction

In Part 3 of the assignment, our focus shifted from security and monitoring to optimizing network performance with the introduction of a Load Balancer. The primary objectives were to evenly distribute requests from designated clients to servers using a round-robin load balancing strategy. This report provides insights into the load balancer's implementation, network behavior, and answers to assignment questions.

Load Balancer Implementation

Load Balancing Logic

We implemented a load balancer that distributes incoming requests from clients (`h1`, `h2`, `h3`) to servers (`h4`, `h5`) in a round-robin fashion. This load balancing strategy ensures that each server receives an equal share of incoming requests.

Server Selection

The load balancer's server selection mechanism alternates between `h4` and `h5` for each incoming request, ensuring a balanced distribution of traffic. ARP requests are processed to facilitate address resolution.

Network Behavior Analysis

Load Balancer Functionality

During our tests, we observed that the load balancer effectively distributes incoming requests to servers, and the round-robin behavior is consistent. The load balancer ensures a fair utilization of server resources.

Impact on Server Utilization

The even distribution of requests contributes to better server resource utilization. As clients initiate requests, the load balancer directs traffic to available servers, preventing any single server from becoming overloaded.